FinGlance

Tecnical Design Document

Gabriel de Almeida Miki

Sumário

INTRODUCTION 03

Business Requirements

Functional

Non Functional

Use Cases

TECHNICAL DESIGN 10

Architectural Pattern

Package Diagram

Composite Structure?

Class Diagram?

DESIGN ELEMENTS 06

TESTING 08

ISSUES 09

Introduction

The FinGlance application is designed to simplify financial tracking for users, providing them with an intuitive and practical tool to manage their finances. The ultimate goal is to offer users a clear and comprehensive overview of their current portfolio status and its evolution over time.

Functional

User Authentication and Authorization
- Users must be able to register and log in using secure credentials.
- Support for third-party authentication providers (e.g., Google, Facebook).

Portfolio Management
- Users can add, edit, and delete various stocks

Dashboard and Visualization
- Show dashboard displaying portfolio key financial metrics.
- Show searched stock price action

Search Capabilities
- Users can search for a specific stock

Profile Management
- Users must be able to edit their profile data

Non Functional

User Authentication and Authorization
- Users must be able to register and log in using secure credentials.
- Support for third-party authentication providers (e.g., Google, Facebook).

Portfolio Management
- Users can add, edit, and delete various stocks

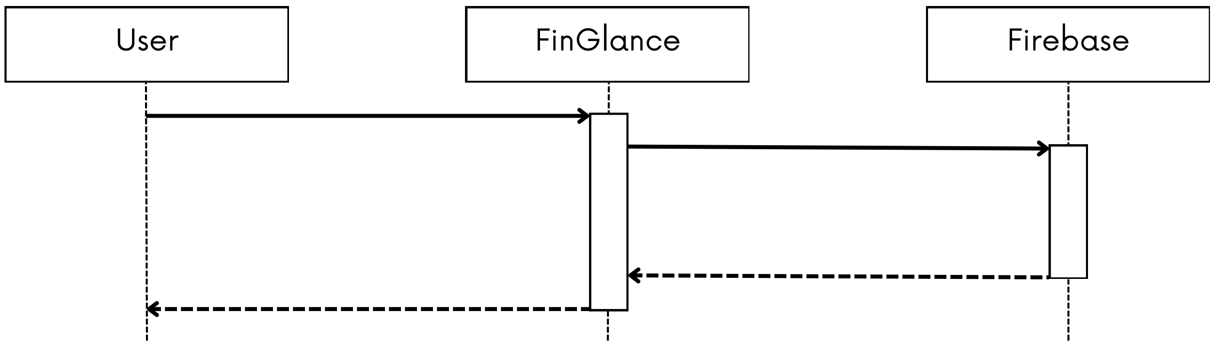
Dashboard and Visualization
- Show dashboard displaying key financial metrics.

Use Case: User Authentication

- Actors:
  - User: The person attempting to log in.
  - Authentication Service: The service (Google, Facebook, or the app's email/password system) that validates user credentials.

Use Case: User Authentication

- Entry condition:
  - The user is on the login screen.
- Flow of Events:
  a. The user selects a login method (Google, Facebook, or email/password).
  b. The user provides the necessary credentials.
  c. The application sends the credentials to the chosen authentication service.
  d. The authentication service validates the credentials.
  e. Upon successful validation, the user is granted access to their portfolio dashboard.
- Exit condition:
  - The user is authenticated and redirected to the portfolio dashboard.
- Exceptions:
  - Invalid credentials: An error message is displayed, prompting the user to try again.
  - Network issues: The user is informed of the connectivity problem and asked to retry.
- Special Requirements:
  - Secure communication (HTTPS) for transmitting credentials.
  - Support for OAuth 2.0 for third-party authentication.
- Nonfunctional Requirements, Constraints:
  - Authentication should be completed within 3 seconds.
  - High availability of authentication services to minimize downtime.



Use Case: Portfolio Management

- Actors:
  - User: The person managing their portfolio.
- Entry condition:
  - The user is logged in and viewing their portfolio dashboard.

Use Case: Portfolio Management

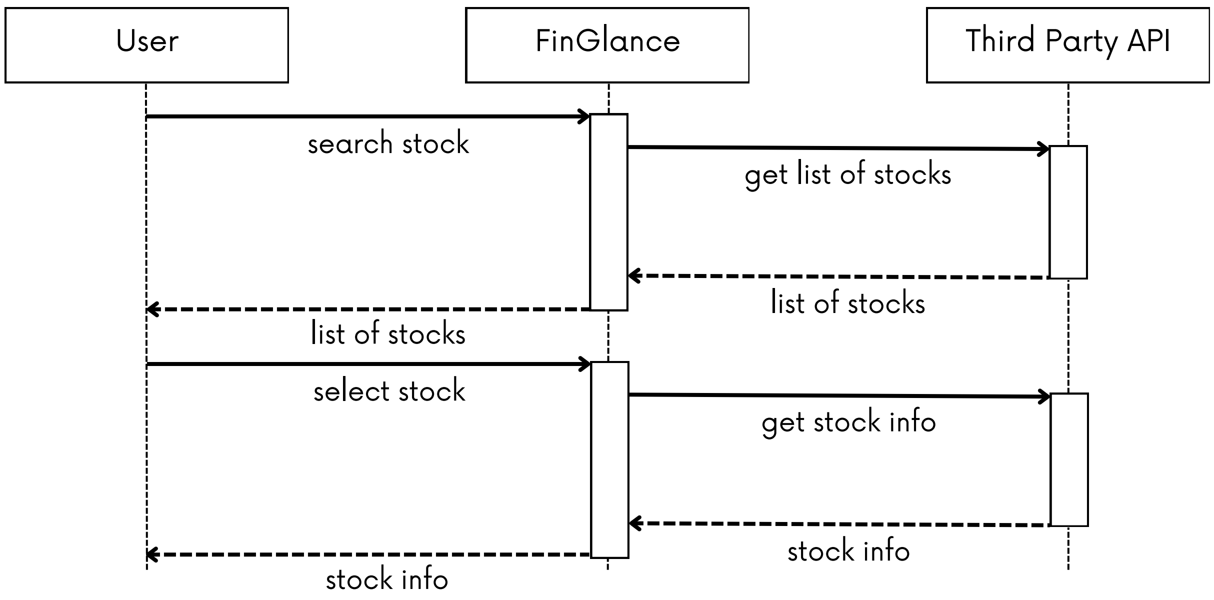
- Flow of Events:
  - The user selects the option to add, delete, or update a stock in their portfolio.
  - For adding a stock:
    - The user searches for a stock by name or symbol.
    - The application displays matching results.
    - The user selects a stock and specifies the number of shares.
    - The application adds the stock to the user's portfolio.
  - For deleting a stock:
    - The user selects a stock from their portfolio.
    - The application prompts for confirmation.
    - Upon confirmation, the stock is removed from the portfolio.
  - For updating a stock:
    - The user selects a stock from their portfolio.
    - The application displays the current details.
    - The user modifies the details (e.g., number of shares).
    - The application updates the stock in the portfolio.
- Exit condition:
  - The user's portfolio is updated to reflect the changes.
- Exceptions:
  - Invalid stock symbol: An error message is displayed.
  - Network issues: Changes are temporarily saved locally and synchronized when connectivity is restored.
- Special Requirements:
  - Real-time data updates for stock prices.
  - Confirmation prompts for delete and update actions to prevent accidental changes.
- Nonfunctional Requirements, Constraints:
  - Portfolio updates should be reflected within 2 seconds.
  - Ensure data consistency and integrity during concurrent modifications.

Use Case: Stock Search and Details

- Actors:
  - User: The person searching for stock information.
- Entry condition:
  - The user is logged in and on the search screen.

Use Case: Stock Search and Details

- Flow of Events:
  - The user enters a stock name or symbol into the search bar.
  - The application retrieves and displays matching stocks.
  - The user selects a stock from the search results.
  - The application displays detailed information about the selected stock, including price action and enterprise details.
- Exit condition:
  - The user views detailed information about the selected stock.
- Exceptions:
  - No matching stocks found: An appropriate message is displayed.
  - Network issues: A message is displayed, and the user can retry the search.
- Special Requirements:
  - Real-time data retrieval for stock prices and details.
- Nonfunctional Requirements, Constraints:
  - Search results should be displayed within 2 seconds.
  - Ensure accurate and up-to-date stock information.

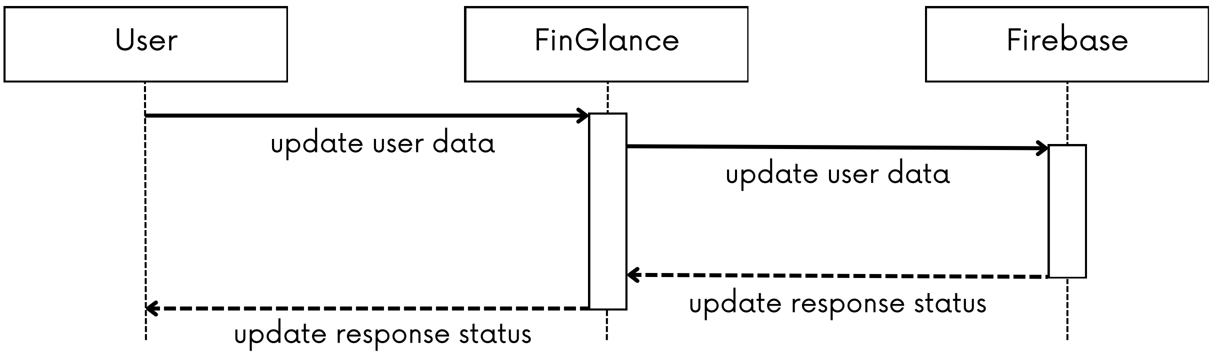


Use Case: Profile Management

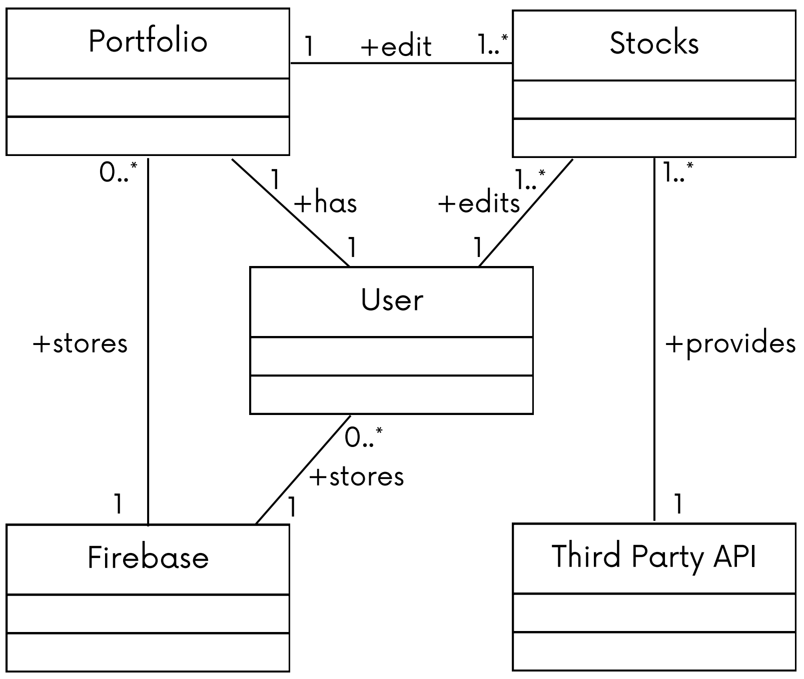
- Actors:
  - User: The person managing their profile.
- Entry condition:
  - The user is logged in and viewing their profile screen.

Use Case: Profile Management

- Flow of Events:
  - The user selects the option to edit their profile.
  - The application displays the current profile details.
  - The user modifies their personal information (e.g., name, email, password).
  - The application validates the changes.
  - Upon successful validation, the application updates the user's profile.
- Exit condition:
  - The user's profile is updated with the new information.
- Exceptions:
  - Invalid input data: Error messages prompt the user to correct the information.
  - Network issues: Changes are temporarily saved locally and synchronized when connectivity is restored.
- Special Requirements:
  - Secure handling of sensitive information (e.g., passwords).
  - Real-time validation of email format and password strength.
- Nonfunctional Requirements, Constraints:
  - Profile updates should be completed within 2 seconds.
  - Ensure data privacy and protection for user information.

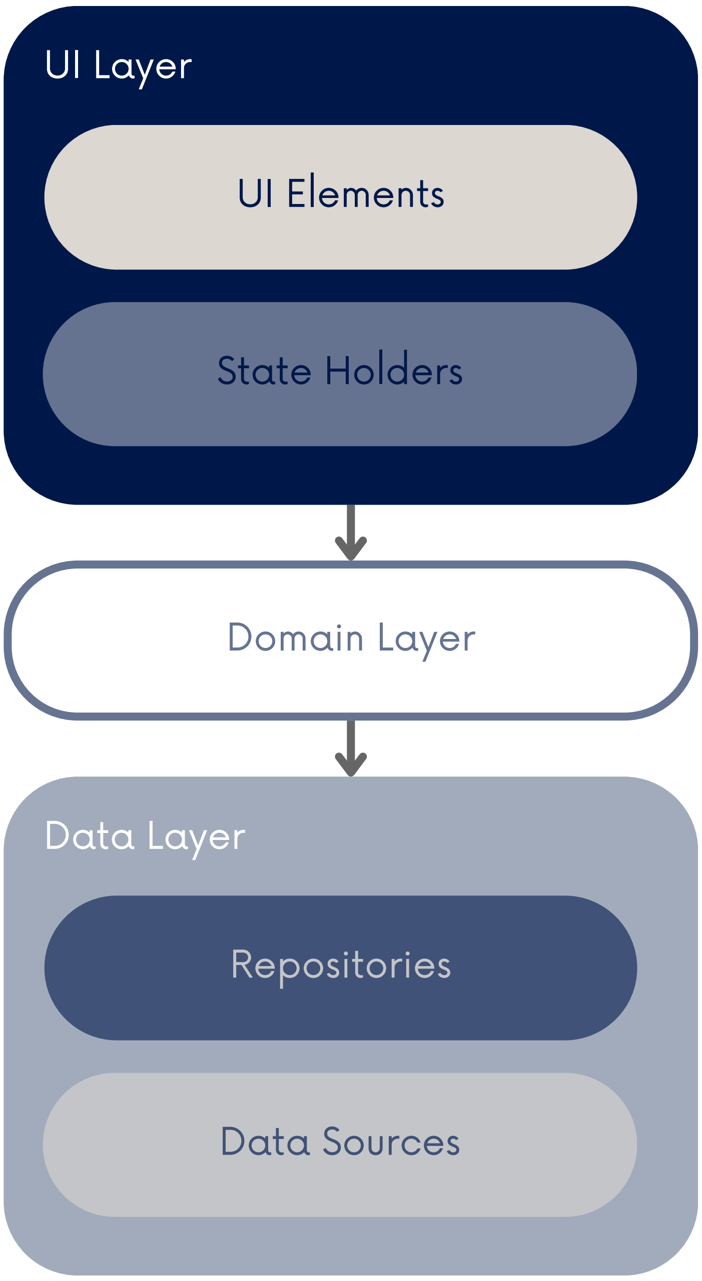


Use Case: Profile Management



Technical Design

The FinGlance application is structured into three primary layers: the User Interface (UI) layer, the Data layer, and the Domain layer. This layered architecture ensures a clear separation of concerns, promoting maintainability, scalability, and testability.



User Interface (UI) Layer

The UI layer is responsible for displaying the application's data on the screen. Whenever data changes, either due to user interactions (e.g., button presses) or external inputs (e.g., network responses), the interface updates to reflect these changes. The UI layer comprises:

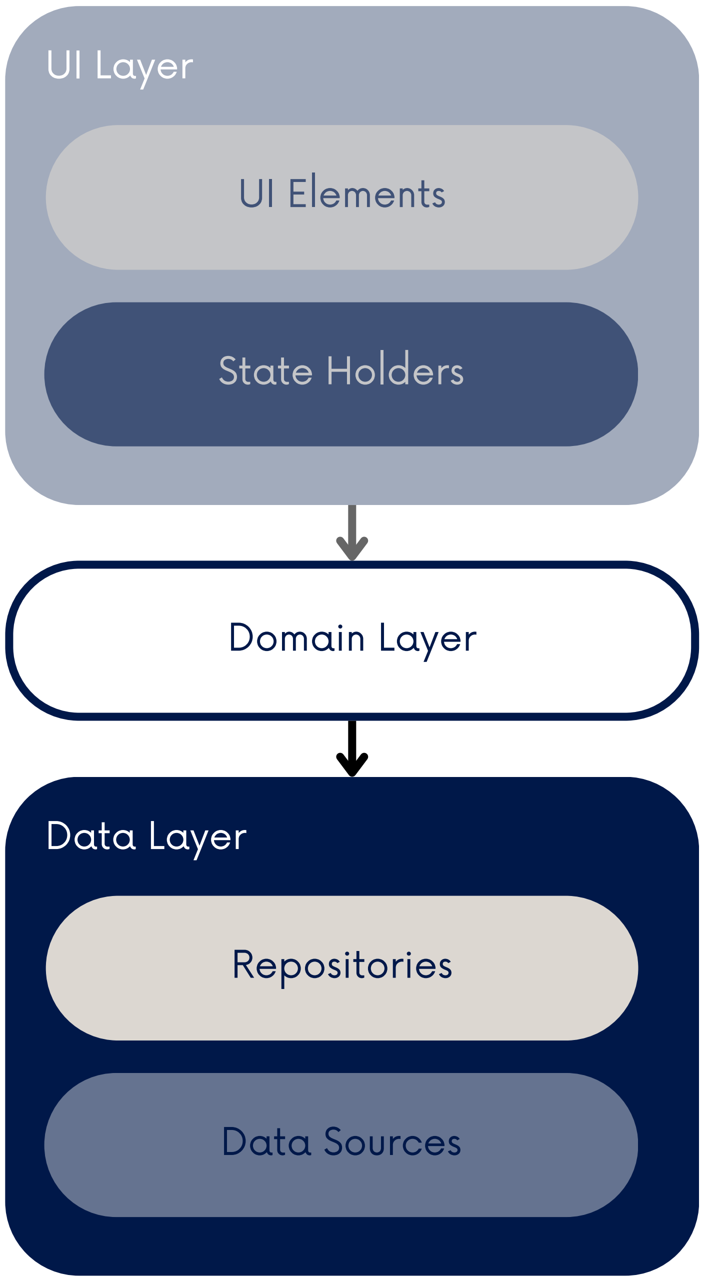
- UI Elements: These render the data on the screen and are created using traditional views or Jetpack Compose functions.
- State Holders: Classes such as ViewModel store data, expose it to the UI, and handle logic related to the UI state.

The primary functions of the UI layer include:

- Rendering dynamic and responsive UI components.
- Handling user interactions and input.
- Observing changes in state and updating the UI accordingly.

Domain Layer

The domain layer serves as an intermediary between the UI layer and the data layer, ensuring that business logic is decoupled from both data handling and UI concerns.



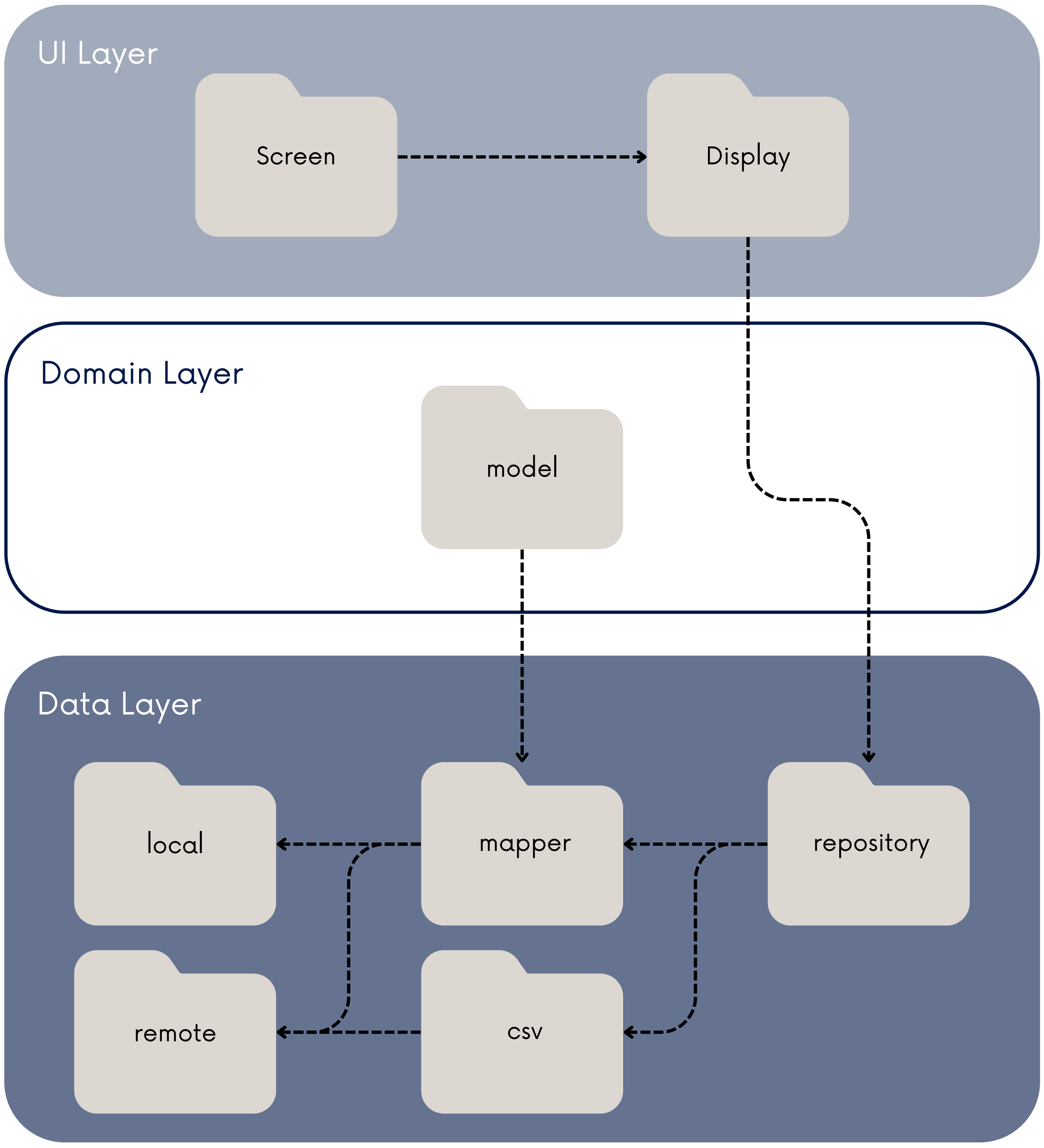
Data Layer

The data layer contains the application's business logic, which adds value by defining rules for data creation, storage, and modification. This layer is primarily composed of repository classes responsible for:

- Data Exposure: Providing data to other parts of the application.
- Data Management: Centralizing and managing data changes.
- Conflict Resolution: Handling conflicts between multiple data sources.
- Abstraction: Abstracting data sources from the rest of the app.
- Business Logic: Containing the core business logic of the application.

Package Diagram

The package diagram provides a high-level view of the FinGlance application's architecture. It illustrates how the application is organized into distinct layers and modules, depicting the dependencies and relationships between these packages. This diagram helps in understanding the separation of concerns and the overall structure of the system.



UI Layer
Screen
Display
Domain Layer
model
Data Layer
local
mapper
repository
remote
csv